Read, Attend and Pronounce: An Attention-Based Approach for Grapheme-To-Phoneme Conversion

Shubham Toshniwal, Karen Livescu

Toyota Technological Institute at Chicago
{shtoshni, klivescu}@ttic.edu

Abstract

We propose an attention-enabled encoder-decoder model for the problem of grapheme-to-phoneme conversion. Most previous work has tackled the problem via joint sequence models that require explicit alignments for training. In contrast, the attention-enabled encoder-decoder model allows for jointly learning to align and convert characters to phonemes. With this approach, we achieve state-of-the-art results on the CMUDict data set with a Word Error Rate (WER) of 23.62%. The performance is comparable even to that of models trained using aligned training data.

Index Terms: grapheme-to-phoneme, LSTM, encoder-decoder, attention

1. Introduction

Grapheme-to-phoneme conversion ("G2P") is the task of converting a from a spelling (a grapheme sequence) to its pronunciation (a phoneme or phone sequence[1]). For example, given a word *able*, the task is to output its pronunciation [EY B AH L]. This conversion is a frequent component of text-to-speech (TTS) and automatic speech recognition (ASR) systems. While static dictionaries exist, they are finite; G2P models are essential for handling out-of-vocabulary words.

One of the main challenges in G2P conversion is that the pronunciation of any grapheme depends on its context. This is exacerbated by the fact that the lengths of the input sequence and output sequence might be different. For example, the pronunciation for *blaze*, a word consisting of 5 characters, is the sequence [B L EY Z] consisting of 4 phones. On the other hand, the very similar grapheme sequence *blase* is pronounced in a very different way, [B L AA Z EY], because of its origin in French. Thus, the alignment between grapheme and phoneme sequences is non-trivial.

A typical approach in previous work involves using joint sequence models, where the model is provided the alignment via some external aligner [1, 2, 3]. However, since the alignment is a latent varible and a means to an end rather than the end itself, it is interesting to consider whether we can do away with such explicit alignments.

Some recent work on the G2P problem has used neural network-based approaches. Specifically, long short-term memory (LSTM) networks have recently been explored for modeling the grapheme and phoneme sequences [4, 5]. LSTMs (and, more generally, recurrent neural networks) can model varying contexts ("memory") and have been successful for a number of sequence prediction tasks. When used in an encoder-decoder approach, as in [5], they in principle require no alignment between the input (grapheme sequence) and output (phoneme sequence) and are therefore quite natural for this task. However, to date the best-performing approaches still use alignments. In this paper we explore an extension of encoder-decoder networks based on an attention mechanism, which has proven useful in other sequence prediction tasks. The attention mechanism endows encoder-decoder networks with the ability to consider "soft" alignments, and to learn these alignments jointly with the sequence prediction task. We show that we can achieve state-of-the-art G2P results with such attention-enabled encoder-decoder models.

[1]We will generally use the term "phoneme" to encompass any subword unit of pronunciation.

2. Model

In this section, we describe the various components of our model.

2.1. Encoder-decoder framework

We briefly describe the *encoder-decoder* framework proposed by [6]. An encoder-decoder model includes an encoder, which reads in the input (grapheme) sequence, and a decoder, which generates the output (phoneme) sequence. In our model, the encoder is a bidirectional long short-term memory (BiLSTM) network; we use a bidirectional network in order to capture the context on both sides of each grapheme. The encoder reads the grapheme sequence, a sequence of vectors $\mathbf{x} = (x_1, \cdots, x_{T_g})$, and outputs a sequence of hidden state vectors, $\mathbf{h} = (h_1, \cdots, h_{T_g})$, given by:

$$\overrightarrow{h}_t = f(x_t, \overrightarrow{h}_{t-1})$$
$$\overleftarrow{h}_t = f'(x_t, \overleftarrow{h}_{t+1})$$
$$h_t = (\overrightarrow{h}_t; \overleftarrow{h}_t)$$

We use separate stacked (deep) LSTMs to model f and f'. A "context vector" c is computed from the encoder's state sequence:

$$c = f(\{h_1, \cdots, h_{T_g}\})$$

This context vector is passed as an input to the decoder. The decoder, $g(.)$, is modeled by another stacked (unidirectional) LSTM, which predicts each phoneme y_t given the context vector c and all of the previously predicted phonemes $\{y_1, \cdots, y_{t-1}\}$ in the following way:

$$p(y_t|\{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, d_t, c)$$

where d_t is the hidden state of the decoder LSTM.

2.2. Attention mechanism

One of the crucial extensions to encoder-decoder models is the use of attention mechanisms to adapt the context vector c for every output label prediction. Here the attention mechanism can be seen as a soft alignment between the grapheme sequence and phoneme sequence. We used the attention mechanism proposed by [7], where the context vector c_t at time t is given by:

$$\begin{aligned} u_{it} &= v^T \tanh(W_1 h_i + W_2 d_t) \\ \alpha_{it} &= \text{softmax}(u_{it}) \\ c_t &= \sum_{i=1}^{T_g} \alpha_{it} h_i \end{aligned}$$

The score α_{it} is a weight that represents the importance of hidden encoder state h_i in generating the phoneme y_t.

3. Experiments

3.1. Data

In order to compare directly with earlier results, we use CMUDict pronouncing dictionary and the same experimental setup as in [1].[2] The dictionary uses 27 graphemes and 39 phonemes. It is split into a 106,837-word train set and a 12,000-word test set. As in previous work, we use 2,670 words sampled from the train set as a validation set, which is used for tuning.

3.2. Evaluation

We evaluate our performance using the standard measures of word error rate (WER) and phoneme error rate (PER). PER is equal to the Levenshtein distance of the predicted phoneme sequence from the ground truth divided by the total number of phonemes in the ground truth. WER is equal to the total number of word errors divided by the total number of words. As in prior work, for words with multiple ground-truth pronunciations, we choose the ground truth that results in the lowest PER.

3.3. Training

Our stacked LSTMs have 3 layers, each with 512 units, and 512-dimensional embeddings. We use minibatch stochastic gradient descent (SGD) together with Adagrad to train our model. We use an initial learning rate of 0.08 (per example) and a batch size of 256. Our model is trained for 100 epochs. To prevent overfitting we: (a) introduce a dropout layer between every pair of consecutive layers of the stacked LSTMs (b) use scheduled sampling [8], with a linear decay, on the decoder side.

3.4. Inference

We use a greedy decoder (beam size = 1) to decode the phoneme sequence during inference.

3.5. Results

We compare the results of our model with some of the best previous results in Table 1. The entries in the first part of the table use an external aligner, while the entries in the second part do not. Our model significantly outperforms the previous models that do not use an external aligner. Moreover, its performance is comparable to models that do use external aligners. We note that, since our model parameters are quite different from those in [5], we cannot necessarily attribute the performance boost entirely to the addition of the attention mechanism.

[2]We are grateful to Stan Chen for providing the data.

Method	**PER (%)**	**WER (%)**
BiDir LSTM + Alignment [5]	5.45	23.55
Kneser Ney 9-gram model [2]	5.88	24.53
Encoder-decoder [5]	7.63	28.61
DBLSTM-CTC [4]	-	25.8
Encoder-decoder + Attention	5.73 ± 0.15	23.62 ± 0.6

Table 1: Comparison of our model's performance with the previous best-performing published results.

4. Conclusion

In this work, we have applied an attention-enabled encoder-decoder model for the problem of grapheme-to-phoneme conversion. The model achieves state-of-the-art WER results on the CMUDict data set, and is comparable to models that use external alignment. In future work, we plan to compare different attention models for this task. In particular, local attention models may be particularly well-suited to exploit the almost monotonic alignments between graphemes and phonemes.

5. References

[1] S. F. Chen, "Conditional and joint models for grapheme-to-phoneme conversion," in *Eurospeech*, 2003.

[2] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Commun.*, vol. 50, no. 5, pp. 434–451, May 2008.

[3] S. Jiampojamarn, G. Kondrak, and T. Sherif, "Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion," in *HLT-NAACL*, 2007.

[4] K. Rao, F. Peng, H. Sak, and F. Beaufays, "Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks," in *ICASSP*, 2015.

[5] K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," in *Interspeech*, 2015.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[7] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. E. Hinton, "Grammar as a foreign language," in *NIPS*, 2015.

[8] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks." *CoRR*, vol. abs/1506.03099, 2015.